Ollscoil Mhá Nuad
Maynooth University

SEMESTER 2
2022-2023

CS608
Software Testing

Prof. O. Conlan, Dr. J. Timoney, Prof. S. Brown

Time allowed: 3 hours

Answer at least ***three*** questions

Your mark will be based on your best ***three*** answers

All questions carry equal marks

Instructions

	Yes	No	N/A
Formulae and Tables book allowed *(i.e. available on request)*		X	
Formulae and Tables book required *(i.e. distributed prior to exam commencing)*		X	
Statistics Tables and Formulae allowed *(i.e. available on request)*		X	
Statistics Tables and Formulae required *(i.e. distributed prior to exam commencing)*		X	
Dictionary allowed *(supplied by the student)*		X	
Non-programmable calculator allowed		X	
Students required to write in and return the exam question paper		X	

[25 marks]

Q1 (a) Explain, with reference to the method **maxv()** defined below, why *exhaustive testing* is infeasible. Consider the test design time and the test execution time in your answer. Note: the maximum array length in Java is Integer.MAX_SIZE-2 (or 2147483645). [5 marks]

The method **int maxv(int[] x)** returns the maximum value in x.

(b) A heating system includes a method **boilerSetting()** as defined below to decide what setting is required for the boiler in a home heating system. Analyse this method in preparation for Combinational Testing using Decision Tables: [15 marks]

- Develop a value line for **t** only
- Define boolean expressions for the causes (parameters **t** & **isOn**)
- Define boolean expressions for the effects (**return value**)
- List all the candidate combinations of causes in a table
- Clearly cross out the infeasible combinations
- Develop a decision table from the table of feasible combinations
- Confirm your decision table by interpreting each rule in turn

Level is defined as follows:

```
public enum Level {NONE, LOW, HIGH};
```

The method boilerSetting() is defined as follows:

boilersetting

```
public static Level boilerSetting(int t, boolean isOn)

A home heating system has a boiler with 3 settings: off,
low, and high. This method determines the setting required
based on the temperature and the on/off switch.

Parameters:
  t - the current temperature (in degrees C)
  isOn - whether the heating system is switched on

Returns:
  If switched off, always return NONE
  Otherwise, return:

    • HIGH - t is below 1 degree
    • LOW - t is at least 1 degree and below 25 degrees
    • NONE - t is at least 25 degrees
```

(c) Based on your answer to Part (b), develop combinational tests using a Decision Table (DT) for the method boilderSetting(). Include three tables in your answer: the Test Coverage Items (TCI), selected data values (use -50 and +50 as sensible very low and very high temperatures), and Test Cases. Review your design: complete the TCI table, show that each test case is covered, and also show that there is no duplicate coverage in your test cases. [5 marks]

[25 marks]

Q2 (a) Compare black-box and white-box testing as follows: [5 marks]

i. What types of errors they are likely to find?
ii. What impact do source code changes have on existing tests?
iii. Can tests be written before the code itself?
iv. What role does the specification play in developing the tests?
v. Is it difficult to measure the coverage that the tests achieve?

(b) The method **Wind.categorise()** describes windspeed by its Beaufort scale name. The Javadoc specification is shown below. [15 marks]

```
public static cs608.Wind.Description categorise(int knots)
```

Categorise the wind conditions based on the wind speed.

Parameters:

`knots` - - the wind speed in Knots

Returns:

ERROR if the speed is less than 0 - sensor error
Otherwise:

- CALM if the speed is less than 1
- LIGHT_AIR if the speed is 1-3
- LIGHT_BREEZE if the speed is 4-6
- GENTLE_BREEZE if the speed is 7-10
- MODERATE_BREEZE if the speed is 11-16
- FRESH_BREEZE if the speed is 17-21
- STRONG_BREEZE if the speed is 22-27
- NEAR_GALE if the speed is 28-33
- GALE if the speed is 34-40
- STRONG_GALE if the speed is 41-47
- STORM if the speed is 48-55
- VIOLENT_STORM if the speed is 56-63
- HURRICANE if the speed is at least 64

The method has already been tested using Black-Box test techniques. On the following page you can see the White-Box coverage achieved by these tests as measured by JaCoCo.

Develop the **additional** tests required to provide full statement coverage (SC) for this method. In your answer make sure to: clearly identify the unexecuted statements, identify and explain the conditions for the inputs required to execute each unexecuted statement, and provide both a completed Test Coverage Items table, and a Test Cases table.

Question continued on the next page.

The following figure shows the measured coverage of existing black-box tests for the method categorise().

Note that lines 35 and 49 are highlighted in red, and lines 34 and 48 in yellow.

```
32.  public static Description categorise(int knots) {
33.     int mps=knots * 5; // convert to 0.1m/s
34.     if (mps<0)
35.        return ERROR;
36.     else if (mps<5)
37.        return CALM;
38.     else if (mps<=15)
39.        return LIGHT_AIR;
40.     else if (mps<33)
41.        return LIGHT_BREEZE;
42.     else if (mps<55)
43.        return GENTLE_BREEZE;
44.     else if (mps<79)
45.        return MODERATE_BREEZE;
46.     else if (mps<107)
47.        return FRESH_BREEZE;
48.     else if (mps<138)
49.        return STRONG_BREEZE;
50.     else if (mps<171)
51.        return NEAR_GALE;
52.     else if (mps<207)
53.        return GALE;
54.     else if (mps<244)
55.        return STRONG_GALE;
56.     else if (mps<=284)
57.        return STORM;
58.     else if (mps<=326)
59.        return VIOLENT_STORM;
60.     else
61.        return HURRICANE;
62.  }
```

(c) Work out the required input parameter values for `x` and `enable` that ensure execution of lines 7 and 9 in the method inRange() shown below. Clearly identify the conditions required to execute each of the lines, and then show how you determine what input parameter values are required to meet these conditions. [5 marks]

```
 1 public boolean inRange(int x, boolean enable) {
 2    int lower=10; int upper=100;
 3    if (enable) upper = 200;
 4    x = x - lower;
 5    upper = upper - lower;
 6    if (x>=0 && x<=upper)
 7       return true;
 8    else
 9       return false;
10 }
```

[25 marks]

Q3 (a) Compare conventional testing and testing 'in class context' as follows: [5 marks]

In your answer, compare testing the static method:

public static boolean isZero(int x)

with testing the instance method:

public void isZero()

in class Numerical defined as

```
class Numerical {
   private boolean x;
   public void setX(int value);
   public boolean getResult();
   public void isZero();
}
```

(b) Develop EP tests 'in class context' for the method Lighting.decide(). [20 marks]

A lighting system can be set to three power settings: 0 is off, 1 is low power, and 2 is full power. Brightness is measured as dark (lux<100), bright (lux>500) or dim (100..500). Readings below 0 indicate a sensor error.

If override is true, then the power setting is always set to full power. Otherwise: (a) in a dark room, full power lighting is required; (b) in a dim room, low power lighting is required; and (c) in a bright room, no lighting is required. When there is a sensor error, the power is set to none.

In your analysis, identify the accessor methods, develop a value line for brightness, and identify the input and output Equivalence Partitions. Also, include three tables in your answer for: the Test Coverage Items, selected data values for brightness, and the Test Cases. Your Test Cases must show the exact sequence of method calls and expected return values. Complete the TCI table, and review your Test Cases.

```
public class Lighting {

   private int power;
   private boolean override;

   // Get the lighting power setting
   public int getPower() {
      return power;
   }

   // Set override
   public void setOverride(boolean value) {
      override = value;
   }

   // Calculate power setting (set power)
   public void decide(int brightness) {
      // code not required to answer the question
   }
}
```

[25 marks]

Q4 (a) You are providing random testing for method **boolean: isSquare(int x)** which returns true if x is a square (e.g. 1, 4, 9, 16, etc). [6 marks]

Describe the three key problems in fully automated random testing, and demonstrate each problem using isSquare() as an example:

(i) the test data problem
(ii) the test oracle problem
(iii) the test completion problem

(b) Develop random tests for **whatSpeed()** using random data selection and the provided Test Coverage Items. [19 marks]

In your answer (hint: copy the incomplete tables first):
(i) Complete the provided Equivalence Value Criteria Table for selecting random values
(ii) Complete the provided Random EP Test Cases Table
(iii) Provide an outline structure of the automated test code
(iii) Describe how this resolves the 3 random test problems

Specification for whatSpeed():

whatSpeed

```
public static CpuCooler.Required whatSpeed(int temp,
boolean variableSpeed)
```

Indicates the speed required for a CPU cooling fan.

Parameters:

`temp` - - the current temperature (in degrees C)

`variableSpeed` - - true if the fan has variable speed control

Returns:

ERROR if the cpu temperature is not within 0..110 degrees
If the fan has variable speed control:
- HIGH if the cpu temperature is above 90 degrees
- LOW if the cpu temperature is above 50 degrees (but not above 90)
- OFF if the cpu temperature is not above 50 degrees

If the fan does not have variable speed control:
- HIGH if the cpu temperature is above 90 degrees
- OFF if the cpu temperature is not above 90 degrees

(Continued on next page.)

(Question 4(b) Continued)

Equivalence Partition Test Coverage Items:

TCI	Inputs	Expected Results
EP1*	temp	Integer.MIN_VALUE..-1
EP2		0..50
EP3		51..90
EP4		91..110
EP5*		111..Integer.MAX_VALUE
EP6	variableSpeed	true
EP7		false
EP8	Return Value	OFF
EP9		LOW
EP10		HIGH
EP11		ERROR

Incomplete Equivalence Value Criteria Table (inputs only)

Parameter	EP	Criteria
temp	Integer.MIN_VALUE..-1	temp <= -1
	0..50	
	51..90	
	91..110	
	111..Integer.MAX_VALUE	
variableSpeed	true	
	false	

Incomplete Random EP Test Cases Table

ID	TCI Covered	Inputs		Expected Results
		temp	variableSpeed	
T1	EP2,6,8	rand(0,50)	true	OFF
T2	EP4,7,10			HIGH
T3	EP3,[6],9			LOW
T4	EP1*,11			ERROR
T5	EP5*,[11]			ERROR